Network Working Group
Request for Comments # 107

NIC # 5806

Output of the Host-Host Protocol
Glitch Cleaning Committee

UCLA
23 March 1971

Robert Bressler
Steve Crocker
William Crowther
Gary Grossman
Ray Tomlinson
James White

Introduction

The Host-Host Protocol Glitch Cleaning Committee met for the second time at UCLA on 8, 9 March 1971, after canvassing the network community. [The results of the (slightly larger) committee's first meeting are documented in RFC #102.] The committee agreed on several modifications to the protocol in Document #1; these modifications are listed below.

At each of the meetings, the committee quickly treated all but one of the extant topics. At the first meeting, the bulk of time was spent considering the interrupt mechanism, and that discussion is summarized in RFC #102. At the second meeting, the committee spent almost all of its time discussing the notion of bytes; this discussion is summarized after the list of modifications.

This RFC entirely supercedes RFC #102, and is an official modification of Document #1. A revision of Document #1 will be written shortly which incorporates the changes listed here.

NCP implementers are to incorporate these changes as soon as possible. NCP implementers also are to estimate on what date their NCP's will be ready and to communicate this estimate to Steve Crocker or his secretary, Bryna Kirstel.

Modifications

I Bytes

Heretofore, a connection has been a bit stream. Henceforth, it is to be a byte stream, with the byte size, S, indicated in the STR command and in each message. The byte size meets the constraint: $1 \leq S \leq 255$.

The choice of the byte size for a connection is a 3rd level protocol issue, but the byte size is constant for the life of a connection. Each message must contain an integral number of text bytes (see below).

II Message Format

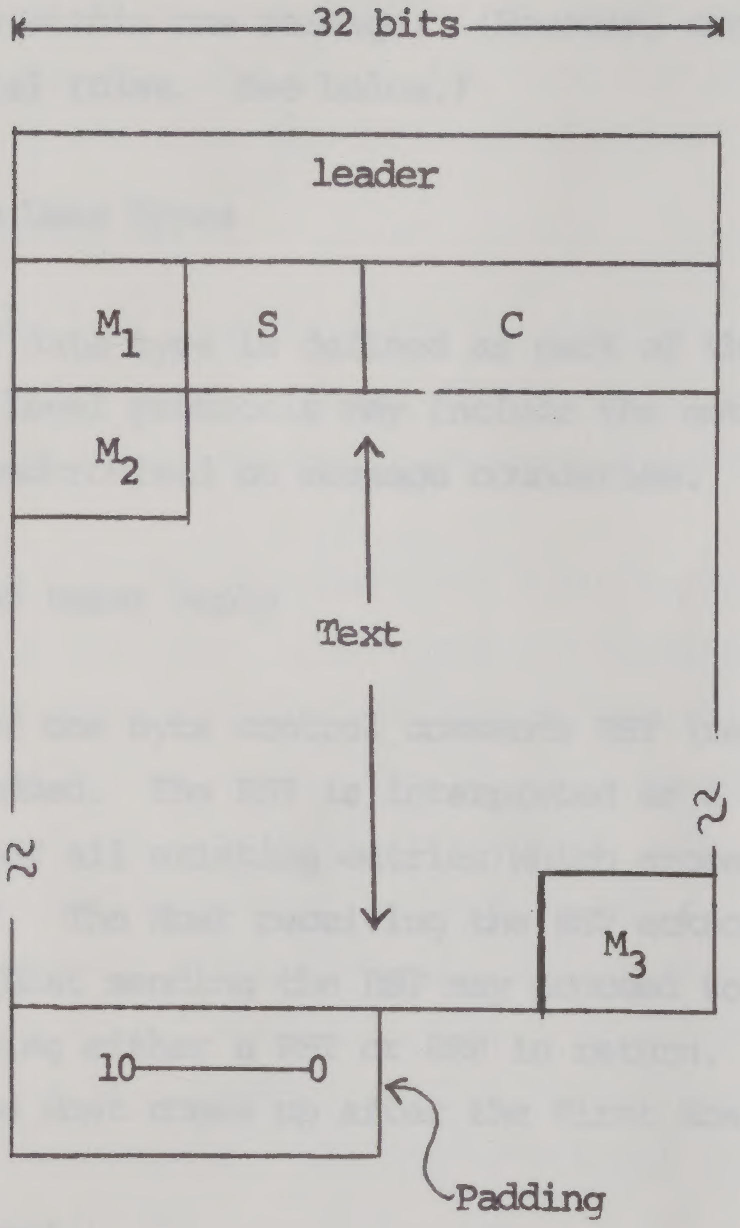
The message format is changed to the format shown in figure 1.

The fields S and C are the byte size and byte count, respectively. The S field is 8 bits wide and must match the byte size specified in the STR which created the connection. The C field is 16 bits long and specifies the number of bytes in the text portion of the message. A zero value in the C field serves no purpose, but is explicitly permitted.

The M_1 and M_2 field are each 8 bits long and must contain zero. The M_3 field is zero or more bits long and must be all zero. The M_3 may be used to fill out a message to a word boundary. It is followed by padding.

The text field consists of C bytes, where each byte is S bits long. The text field starts 72 bits after the start of the message.

The partitioning of a byte stream into messages is an artifact required by the subnet. No semantic content should be attached to message boundaries. In particular,


32 bits
leader
M_1
S
C
M_2
Text
M_3
10——0
Padding

Typical Message
Figure 1

1. A message with a zero value for C has no meaning, although it is legal and it does use up resource allocation. (See Flow Control below.)
2. A receiver may not expect to see 3rd level control information synchronized with message boundaries. Particularly, if the notion of record is defined for a connection, the receiver must expect multiple records and/or record fragments within one message. (However, control messages obey special rules. See below.)

III Message Data Types

No notion of data type is defined as part of the 2nd level protocol. 3rd level protocols may include the notion. Data types cannot be synchronized on message boundaries.

IV Reset and Reset Reply

A new pair of one byte control commands RST (reset) and RRP (reset reply) are added. The RST is interpreted as a signal to purge the NCP tables of all existing entries which arose from the Host which sent the RST. The Host receiving the RST acknowledges by returning a RRP. The Host sending the RST may proceed to request connection after receiving either a RST or RRP in return. An RST is returned if the second Host comes up after the first Host.

V Flow Control

The flow control techniques are changed in two ways. First, the Cease mechanism is discontinued. The 10HI and 11HI messages will no longer be recognized by the Imps, and the Imps will no longer generate the 10IH, 11IH or 12IH messages.

Second, the allocation mechanism now deals with two quantities, bits and messages. The receiver allocates each of these quantities separately. The sender and receiver each must maintain a 16 bit unsigned counter for messages and a 32 bit unsigned counter for bits. When sending a message, the sender subtracts one from the message counter, and the text length from the bit counter. The receiver decrements his counters similarly when receiving the message. The sender is prohibited from sending if either counter would be decremented below zero. Similarly, the receiver is prohibited from raising the current message allocation above $2^{16}-1$, or the current bit allocation above $2^{32}-1$.

The *text length* of a message is the product of S, the byte size, and C, the number of bytes. These values always appear in the first part of the message, as described under Message Format.

The ALL, GVB, and RET commands are modified to treat two quantities. Their formats are given under Control Commands, below. The GVB command is further modified to make it possible to ask for none of the allocation to be returned. The new GVB command has four eight bit fields. The first two fields are the op code and the link, as before. The next two fields contain numbers f_m and f_b which control how much of message and bit allocation are to be returned. *Each of these numbers is interpreted as "the number of 128ths of the current allocation" to be returned if it is in the range zero to 128, and is to be interpreted as "all of the current allocation" if it is in the range 128 to 255.*

VI Control Messages

The control link is changed to link 0; link 1 is not to be used. The old and new protocols may therefore coexist.

Second, the allocation mechanism now deals with two quantities, bits and messages. The receiver allocates each of these quantities separately. The sender and receiver each must maintain a 16 bit unsigned counter for messages and a 32 bit unsigned counter for bits. When sending a message, the sender subtracts one from the message counter, and the text length from the bit counter. The receiver decrements his counters similarly when receiving the message. The sender is prohibited from sending if either counter would be decremented below zero. Similarly, the receiver is prohibited from raising the current message allocation above $2^{16}-1$, or the current bit allocation above $2^{32}-1$.

The *text length* of a message is the product of S, the byte size, and C, the number of bytes. These values always appear in the first part of the message, as described under Message Format.

The ALL, GVB, and RET commands are modified to treat two quantities. Their formats are given under Control Commands, below. The GVB command is further modified to make it possible to ask for none of the allocation to be returned. The new GVB command has four eight bit fields. The first two fields are the op code and the link, as before. The next two fields contain numbers f_m and f_b which control how much of message and bit allocation are to be returned. *Each of these numbers is interpreted as "the number of 128ths of the current allocation" to be returned if it is in the range zero to 127, and is to be interpreted as "all of the current allocation" if it is in the range 128 to 255.*

VI Control Messages

The control link is changed to link 0; link 1 is not to be used. The old and new protocols may therefore coexist.

Messages sent over the control link have the same format as other regular messages, as described above under Message Format. The byte size field must contain the value 8.

Control messages may not contain more than 120 bytes of text; the value in the byte count field is thus limited to 120. This limitation is intended to help smaller Hosts.

Control messages must contain an integral number of control commands. Control commands, therefore, may not be split across control messages.

VII Link Assignment

The links are now assigned as follows:

0	control link
1	old protocol's control link - to be phased out
2 - 31	links for connections
32 - 190	reserved -- not for current use
191	to be used only for measurement work under direction of the network measurement center (UCLA)
192- 255	available for any private experimental use

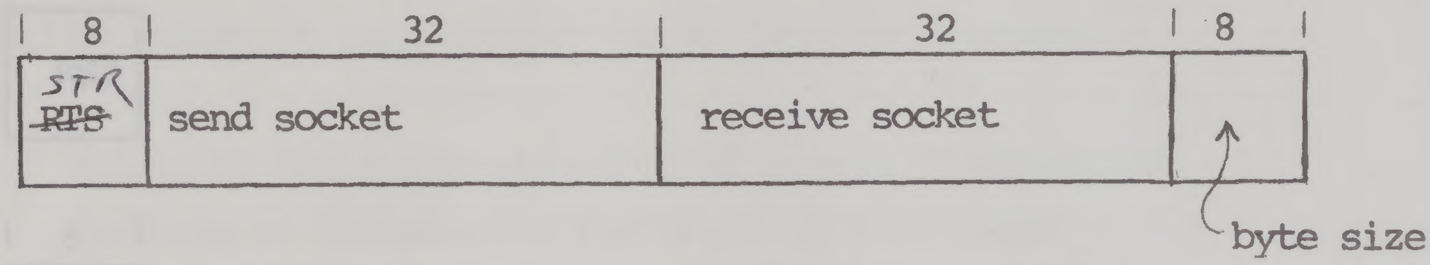
VIII Fixed Length Control Commands

The ECO, ERP and ERR commands are now fixed length. The ECO and ERP are now 16 bits long -- 8 bits of op code and 8 bits of data. The ERR command is now 96 bits long -- 8 bits of op code, 8 bits of error code, and 80 bits of text. 80 bits is long enough to hold the longest non-ERR control command.

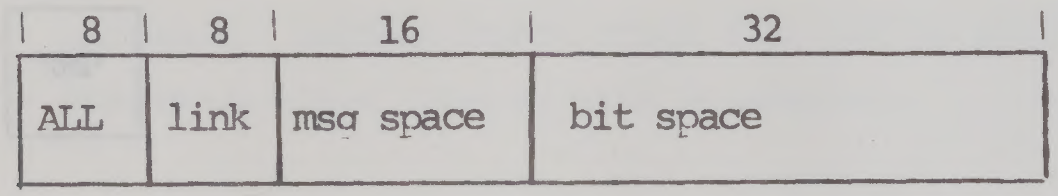
IX Control Command Formats

As mentioned above, the formats of the STR, ALL, GVB, RET, ECO, ERP and ERR commands have changed; and the commands RST and RRP have been added. The formats of these commands are given here.

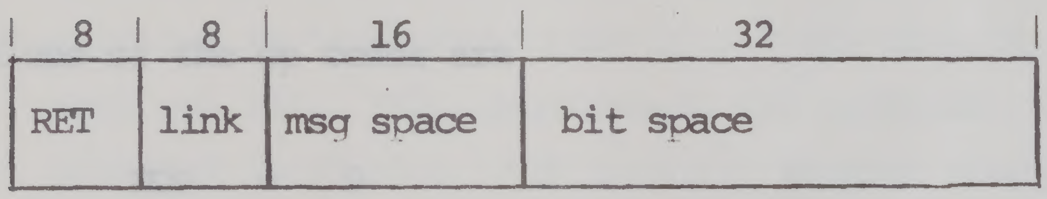
1.


8
32
32
8
STR
~~RTS~~
send socket
receive socket
byte size

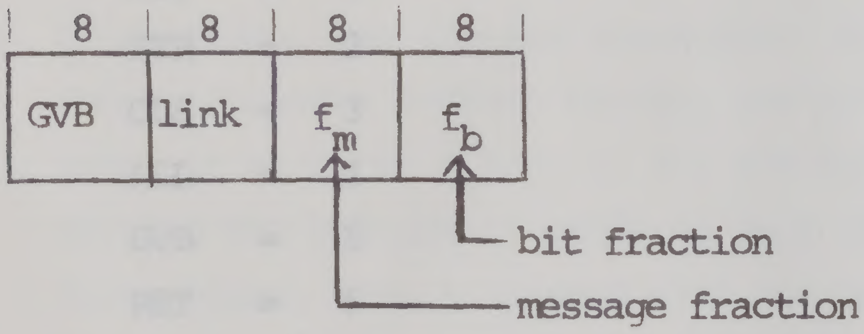
2.


8
8
16
32
ALL
link
msg space
bit space

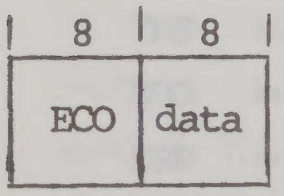
3.


8
8
16
32
RET
link
msg space
bit space

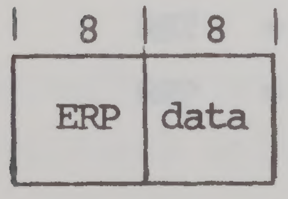
4.

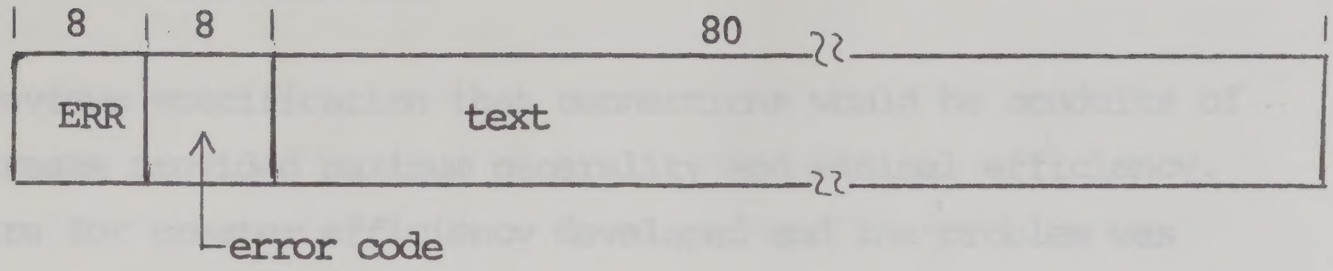

8
8
8
8
GVB
link
f_m
f_b
bit fraction
message fraction

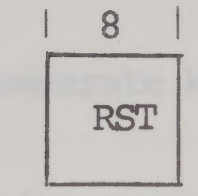
5.

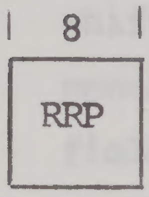


6.



7. 
8
8
80
ERR
text
error code

8. 

9. 

The values of the op codes are

NOP = 0
RTS = 1
STR = 2
CLS = 3
ALL = 4
GVB = 5
RET = 6
INR = 7
INS = 8
ECO = 9
ERP = 10
ERR = 11
RST = 12
RRP = 13

Discussion on Byte Streams

The previous specification that connections would be conduits of bit streams provided maximum generality and minimal efficiency. Pressure for greater efficiency developed and the problem was examined.

Two separate kinds of inefficiency arose from bit streams.

1. Receiving Hosts were required to engage in expensive shifting to concatenate the texts of successive messages. Sending Hosts often also had to shift text fields to align them on word boundaries.

2. Sending NCP's were prohibited from hanging onto *any* text for an indefinite time if it were possible to send even one bit. This requirement was necessary to prevent possible deadlocks. For example, suppose processes A and B have a conversation in progress over a pair of connections, one in each direction. Also suppose that these processes produce exactly one bit of output for each bit of input. Then if A's NCP fails to send a waiting bit because it wants to pack it together with later output from A, then B will not be able to output and neither will A. It is clear then, that unless there is some guarantee that the data in the sending NCP's buffers are not crucially needed on the receive side, the sending NCP must assume otherwise and transmit any waiting data as soon as it is able.

These considerations led to the notion of a "transmission unit," whose existence would be known to the NCP's. The question then became what were typical and/or possible transmission unit sizes. For

character-oriented interaction, 8-bit transmission units seemed reasonable. For line-oriented interaction, the transmission unit might best be the line itself, and therefore variable length; alternatively, it might be best to consider the transmission unit to be a character. For file transfer, it might be desirable for the transmission unit to be a multiple of the word lengths of both machines; however, the last part of the file may not form a whole transmission unit if the transmission unit is too large. The consensus became that the transmission unit should not be divisible under any circumstances, and should, therefore, be fairly small. The notion of transmission unit thus seems to be synonymous with the notion of byte, and the term transmission unit was dropped.

Subsequent discussion of the deadlock and wakeup aspect revealed that there may be two byte sizes associated with a single connection:

1. Transmission from the sending process to the sending NCP is in bytes of size S. The sending NCP must send a message whenever the link is unblocked, the message counter is at least 1, the bit counter is at least S, and at least S bits of text are ready. The message must contain an integral number of bytes.

2. At the receiving side, there may be a different byte size R for transmission from the receiving NCP to the receiving process. An example of where R ≠ S, is suggested by UCSB which is providing a file system for transparently storing binary files. It is reasonable that a using Host might send with 36 bit bytes, while the UCSB file system might want to receive 32-bit increments.

It is clear that from a network protocol point of view, only the byte S is relevant, and this is quantity which is communicated in the STR command and in every message. The choice of the byte size R is up

to the receiving user, and its meaning is how often the receiving NCP should wakeup the receiving process. It may also happen that a receiving process has an agreement with the receiving NCP which is more complex than "please wake me every R bits;" for example, the NCP might scan for new-line characters before waking up the receiving process.

In the new protocol, it is the option of the receiver to refuse a request for connection on the basis of the proffered byte size. Conceptually, we imagine that NCP's are capable of handling all byte sizes, and that such a choice would be up to the third level programs (user programs, loggers, telnets, etc.) Some Hosts, small ones in particular, may know enough about their third level programs to restrict the variety of byte sizes which can be sent or received. While it is a matter of local policy, the committee strongly suggests that NCP's be capable of handling all byte sizes. One of our committee, moreover, feels strongly that NCP's should be written to be able to receive all byte sizes S and provide for different byte sizes R for transmission to the user process.